PERGAMON

Applied Mathematics Letters 12 (1999) 57–62

Applied Mathematics Letters

A Box Spline Subdivision Pyramid Algorithm

D. J. HEBERT
Department of Mathematics, University of Pittsburgh
Pittsburgh, PA 15260, U.S.A.

(Received December 1997; accepted January 1998)

Communicated by R. Janssen

Abstract—Using the refinement coefficients of the ZP-element box spline to form a low pass filter mask, and a subdivision difference as a high pass filter, we scan an image along a triangular Sierpinski path to create an efficient one-dimensional pyramid algorithm for a two-dimensional nonseparable wavelet-like transform based on the quincunx multiresolution analysis.

1. INTRODUCTION

Wavelets based on four-directional box splines are of interest in image processing as natural extensions of one-dimensional spline wavelet methods [1]. The symmetries of these splines and their directional derivatives also offer the possibilities of improvements in low pass filtering for image compression and in direction selectivity for edge and feature detection [2]. Calculations with box splines, however, have not appeared as practical alternatives to standard methods such as pyramid algorithms based on separable biorthogonal wavelet filters [3,4]. Indeed, the first construction of a four-directional box spline wavelet was obtained only recently by Chui *et al.* [5]. More recently, Ron and Shen [6], as an application of their definitive study of affine frames [7], have constructed box spline wavelets of small support which are suitable for computations and for edge detection. The stage is set for efficient box spline wavelet calculations.

The purpose of the present article is to show how to construct a pyramid algorithm in which the low pass filters are the lowest order four-directional box splines, the Zwart-Powell (ZP) elements [8,9]. The high pass filters may be regarded as discrete directional wavelets. This algorithm is related to the box spline computations suggested by Diamond *et al.* [10], but is based on the quincunx multiresolution analysis used by Chui *et al.*, previously studied by Cohen and Daubechies, and others (cf. [11,12]).

One novelty of the pyramid algorithm given here is the extreme simplicity of calculation obtained by processing an image which has been scanned along a Sierpinski path [13] to form a linear stream of pixel intensities systematically filling all the triangles of a binary triangulation tree (cf. [14,15]) of the type studied in n dimensions by Maubach [16] who gives complete references. The Sierpinski scan traverses a Hamiltonian cycle in the adjacency graph at each tree level, and interlaces corresponding levels of the related Cartesian and quincunx lattice quadtrees [17]. Thus, we obtain a one-dimensional algorithm with two-dimensional nonseparable filter masks.

0893-9659/98/$ - see front matter Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX
PII: S0893-9659(98)00149-9

For computations, the triangular scanning offers efficiencies: the Hamiltonian cycles are indexed by the binary integers which locate binary tree position; all the vertices of the visited triangles are simultaneously represented in as binary matrices, n-tuples of 4-ary digits, and pairs of dyadic rational coordinates; all box spline coefficients can be computed by accumulation of local vertex contributions along the triangular path; Cartesian and quincunx squares are visited twice on the Sierpinski path (each of two triangular halves is visited once) in such a way that data from one triangular half may be pushed to a stack at the first visit and popped for combination with data from the other half at the second visit. This stack structure makes possible the efficient construction of edge and feature detection directional wavelets as differences (which are also directional derivatives) of spatially adjacent splines. For details and computational examples, the reader is referred to [14,18,19].

1.1. Box Splines

We start with some results based on notation of the book by de Boor *et al.* [20], a recent treatment of the subject of box splines with full references. Let $I_0 = [0, 1)$ and suppose that Ξ is an $s \times n$ matrix. The simplest definition the box spline states that $\mathcal{M}_\Xi$ is the distribution determined by the formula $\langle \mathcal{M}_\Xi, \varphi \rangle = \int_{\mathbf{I}_0^n} \varphi(\Xi t)\, dt$ for $\varphi \in C(\mathbf{R}^s)$. In the present article, we are interested in the ZP element box spline for which $s = 2$, $n = 4$, and $\Xi = \Xi_4. = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_1 + \mathbf{e}_2, \mathbf{e}_1 - \mathbf{e}_2]$, where $I = [\mathbf{e}_1, \mathbf{e}_2]$ is the 2×2 identity matrix. In this case, the distribution $\mathcal{M}_\Xi$ has a density function $\mathcal{B}$ which is piecewise quadratic, continuously differentiable, and has compact octagonal support. The second derivative is discontinuous only on edges and diagonals of Cartesian squares, and a surface plot approximates that of the two-dimensional Gaussian density. The ZP element satisfies a dilation (refinement) equation and is the scaling function for a multiresolution analysis of $L^2(\mathbf{R}^2)$ based on scaling by the *quincunx matrix* $M = [\mathbf{e}_1 + \mathbf{e}_2, \mathbf{e}_1 - \mathbf{e}_2]$ (cf. [5]). The scaled and shifted box splines, expressed as $\mathcal{B}^n(x) = 2^n \mathcal{B}(M^n x)$ and $\mathcal{B}^{n,k}(x) = 2^n \mathcal{B}(M^n(x - k))$, are used to compute quasi-interpolants of functions $f : M^{-n}\mathbf{Z}^2 \to \mathbf{R}$ defined on the scaled integer lattice by the formula

$$(\mathcal{B}^n * f)(x) = \sum_{j \in M^{-n}\mathbf{Z}^2} \mathcal{B}^n(x - j) f(j), \qquad x \in \mathbf{R}^2.$$

For computer calculations, the box splines are approximated by discrete box splines: For $\Xi \in \mathbf{Z}^{s \times n}$ and $h \in 1/\mathbf{N}$, the discrete box spline b_Ξ^h is defined by the formula: $\langle b_\Xi^h, \varphi \rangle = h^n \sum_{\alpha \in \mathbf{I}_h^n} \varphi(\Xi\alpha)$ for $\varphi \in C(\mathbf{R}^s)$, where $I_h^n = I_0^n \cap h\mathbf{Z}^n$. If we write $b^h = b_{\Xi_4}^h$, the main results of interest are the refinement equation $b^{hh'} = b^h * b^{h'}(\cdot/h)$ and the factorization $b^{1/2} = p * p(M^{-1}\cdot)$, where $p = b_I^{1/2}$. From the refinement equation, we obtain a discretization of the box spline interpolation formula: if a^l is a function defined on a coarse grid $M^{-l}\mathbf{Z}^2 = (1/2^l)M^l\mathbf{Z}^2$, where $l < 2m$, then $(b^{1/2^m} * a^l)(k) = \sum_{j \in M^{-l}\mathbf{Z}^2} b^{1/2^m}(k - j) a^l(j)$ for $k \in (1/2^m)\mathbf{Z}^2$. The refinement equation also says

$$b^{1/2^m} = b^{1/2} * b^{1/2^{m-1}}(2\cdot) = b^{1/2} * b^{1/2}(2\cdot) * \cdots * b^{1/2}\left(2^{m-1}\cdot\right) = *_{j=0}^{m-1} b^{1/2}\left(2^j\cdot\right),$$

and it follows that

$$b^{1/2^m} = *_{j=0}^{m-1}\left(p * p\left(M^{-1}\cdot\right)\right)\left(2^j\cdot\right) = *_{j=0}^{m-1}\left(p\left(M^{2j}\cdot\right) * p\left(M^{2j-1}\cdot\right)\right) = *_{l=0}^{2m-1}\left(p\left(M^{l-1}\cdot\right)\right).$$

This formula is the basis of a simplicity of calculation of the discrete interpolation of a function a^l defined on a coarse grid and indeed is a form of the subdivision algorithm given in [20]. In image processing, a^l represents a low resolution image and $b^{1/2^m} * a^l$ is a fine scale rendering of a smoothing of this image.

We focus on the scaled and centered box splines defined as $b_l(x) = b_{\Xi_4}^{1/2}((1/2)M^l x + c)$, where $c = (1/4)(3\mathbf{e}_1 + \mathbf{e}_2)$ and $p_l(x) = b_I^{1/2}((1/2)M^l x + (1/4, 1/4))$, whose nonzero coefficients are computed by the simple masks:

$$b_l : \frac{1}{16}\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \qquad p_l : \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

centered at the origin with values on the lattice $L^l = M^{-l}((1/2, 1/2) + \mathbf{Z}^2)$. The mask b_l is the normalized refinement mask for the ZP element scaled by the diagonal matrix $2I$. The factoring of $b^{1/2}$ becomes $b_l = p_l * p_{l-1}$.

1.2. Lattice Origins, Vertices, and Centers

For each positive integer l, we define the set V_l of level l lattice points in the unit square $\mathbf{I}^2 = [0, 1]^2$ as

$$V_l = M^{-l}\mathbf{Z}^2 \cap \mathbf{I}^2.$$

For even l, the level l lattice points are the vertices of Cartesian squares; for odd l the squares whose vertices are the level l lattice points are called quincunx squares. The set of centers of level l lattice squares is defined as

$$C_l = M^{-l}\left(\left(\frac{1}{2}, \frac{1}{2}\right) + \mathbf{Z}^2\right) \cap \mathbf{I}^2.$$

It may be noted that $V_l \cup C_l = V_{l+1}$, so that $V_{l+1} = V_0 \cup \bigcup_{k \le l} C_k$. The elements of C_l can be written in the form

$$x = x(\mathbf{j}) = \left(\frac{1}{2}, \frac{1}{2}\right) + \sum_{k=1}^{m} \frac{1}{2^{k+1}} M^b \sigma_{j_k},$$

where $b \in \{0, 1\}$, $l = 2m - b$, $\mathbf{j} = (j_1, \ldots, j_m)$ is a list of four-ary digits called the lo code of x, σ_{j_k} is an ordered pair of signs: $\sigma_0 = (-1, -1)$, $\sigma_1 = (-1, 1)$, $\sigma_2 = (1, -1)$, $\sigma_3 = (1, 1)$. For even l, the elements of C_l are called *Cartesian Lattice Origin (clo) points* of depth $m = l/2$, while for odd l, the elements of C_l are called *Quincunx Lattice Origin (qlo) points* of depth $m = (l + 1)/2$. The clo points form nodes of a quadtree (4-ary tree) whose root is $(1/2, 1/2)$ while the qlo points form the intersection of $\mathbf{I}$ with the nodes of a quadtree whose root is $(1/2, 1/2)$. The list $\mathbf{j}$ is equivalent to the quadtree locational code whose computational advantages are well known (cf. [17]).

In terms of the present notation, the factorization of b_l has a simple geometric formulation

$$b_l(v) = \sum_{i,j} p_l(k_{ij} - v_i) p_{l-1}(v_i - v),$$

where $v_i = v + M^{-l+1}\sigma_i$ and $k_{ij} = v_i + M^{-l}\sigma_j$.

1.3. Binary Triangulations

We construct a binary tree of triangles as follows: first, two level 0 triangles T_0 and T_1 are obtained by bisecting the unit square $\mathbf{I}^2 = [0, 1]^2$ along the diagonal line joining the points $(0, 0)$ and $(1, 1)$. If T_i is a triangle of level l, then the children of T_i are level $l + 1$ triangles T_{2i} and T_{2i+1}, obtained by bisecting the longest edge of T_i. After a finite number of steps, we obtain a binary tree with root $\mathbf{I}^2$ and with two subtrees of triangles with roots T_0 and T_1. At level l all triangles have the same congruence class and form a uniform triangulation of the unit square in which the vertices are the elements of V_l.

At each level, there is a natural ordering of the two children of a given triangle given geometrically by counterclockwise motion around a common Cartesian lattice origin. Making use of this ordering of children, we may establish a natural linear ordering of the triangles of level l indexed by the positive integers called cycle indices whose l binary digits locate the triangles as elements of a binary tree. For odd values of l, the path connecting the centers of these leaf triangles is known as a *Sierpinski path*, cf. [13]. For any level l, we shall call the path through the triangles of l a *level cycle* or a *Hamiltonian cycle*. Algorithms for traversing these level cycles along with more detailed discussion of the above properties and computational efficiencies may be found in [14,18,19].

1.4. Triangle Vertex Codes and Wavelet Coefficients

The set V_l of level l lattice points coincides with the set of all vertices of level l triangles and includes all the lattice origin sets C_k for $k < l$. The vertices of a triangle T of level $l = 2m$ ($l = 2m - 1$) are labeled $v_0(\mathrm{T})$, $v_1(\mathrm{T})$, and $v_2(\mathrm{T})$, where $v_0(\mathrm{T})$, a lattice origin in the set C_{l-1}, is the vertex opposite the longest edge and where $v_1(\mathrm{T})$ and $v_2(\mathrm{T})$ are ordered by clockwise (counterclockwise) orientation around v_0 when l is even (odd). The midpoint of the edge joining $v_0(\mathrm{T})$ and $v_1(\mathrm{T})$, called $m_1(\mathrm{T})$ is a lattice origin point in the set C_{l+1} which we designate as the representative lo point of the triangle T. The midpoint $m_3(\mathrm{T})$ of the side opposite v_0 is a lattice origin point in the set C_l and is the common vertex for the children of the triangle T (see Figure 1a). By connecting the points $m_1(\mathrm{T})$ as we traverse a level cycle, we form a nonredundant cyclic path called a level cycle through the lattice origins in C_{l+1}. For image processing, this provides a path through pixel centers called a *Sierpinski scan path*. Simultaneously the points $m_3(\mathrm{T})$ traverse a path through the lattice origins in C_l. In this manner, the clo and qlo quadtrees are interlaced by the path. All vertex calculations may be obtained by simple manipulation of the lo codes for the triangle vertices and representative points.

For a triangle T of level l, consider the box spline $f^{l,\mathrm{T}}$ (associated with V_l) centered at $m_1(\mathrm{T})$ defined by $f^{l,\mathrm{T}}(M^{-l}k) = b^l_{\mathrm{A}}(M^{-l}k - m_1(\mathrm{T}))$ where $\mathrm{A} = [\mathbf{a}, \mathbf{a}, \mathbf{b}]$, $\mathbf{a} = v_0(\mathrm{T}) - v_1(\mathrm{T})$, and $\mathbf{b} = v_0(\mathrm{T}) - v_2(\mathrm{T})$. A function a^{l-1} defined on the grid C^l is interpolated at a point $m_1(\mathrm{T})$ in C^{l+1} by the $f^{l,\mathrm{T}}$ subdivision formula $s^l(m_1(\mathrm{T})) = \langle f^{l,\mathrm{T}}, a^{l-1}\rangle$. The number $f^{l,\mathrm{T}}$ will serve as a "wavelet" coefficient in the following algorithm (see Figure 1b for the flow of data to $s^l(m_1(\mathrm{T}))$).

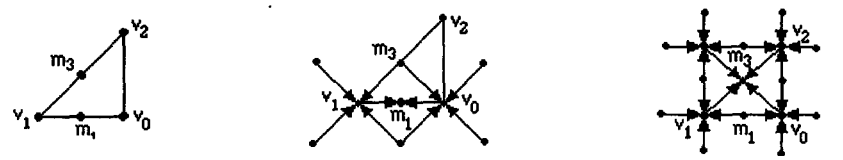

Figure 1.

2. A BOX SPLINE SUBDIVISION PYRAMID

We begin with an image of size $2^m \times 2^m$ whose pixel intensities a_i^{2m-1} are located at the Cartesian lattice origin points $m_1(i) = m_1(\mathrm{T}_i^{2m-1})$ of depth m, where t_i^{2m-1} is the triangle of binary tree level $2m - 1$ indexed by the cycle index i. For each level l, and for each level l cycle index i, we compute the smoothing coefficient (or level $l - 1$ resolution image intensity), a_i^{l-1} as the box spline interpolation $a_i^{l-1} = (b^{l+1} * a^l)(m_3(i)) = \langle b_{l+1,i}, a^l\rangle$, where $b_{l+1,i} = b_{l+1}(\cdot - m_3(i))$ (see Figure 1c for the flow of data to a_i^{l-1} at $m_3(i)$). For points near the boundary, we extend the image intensities by reflection in the boundary. When the computations of $\{a_i^{l-1}\}$ are complete we compute, for the index $m_1(i)$, the $f^{l,\mathrm{T}}$ subdivision interpolation from lower resolution intensities as $s_i^l = s^l(m_1(\mathrm{T})) = \langle f^{l,\mathrm{T}}, a^{l-1}\rangle$. The difference $d_i^l = a_i^l - s_i^l$ is the "wavelet" coefficient at level l to be saved for later reconstruction of the image by the formula $a_i^l = s_i^l + d_i^l$.

Thus, we may construct a pyramid algorithm for calculating the wavelet coefficient hierarchy $a_0^0, d_0^0, d_0^1, d_1^1 d_0^2, \ldots, d_0^{2m-1}, \ldots, d_{2^{2m}-1}^{2m-1}$, and a reconstruction algorithm for regaining the original image.

PROPOSITION. *The pyramid algorithm described above may be calculated by repeated traversal of the level cycle path at each level, accumulating a_i^{l-1} and s_i^{l-1} locally at the vertices of triangles in the path.*

Pseudo-Code

We form an array of size $2^m + 1 \times 2^m + 1$ which we identify with the set V_{2m-1} of vertices of triangles of level $2m-1$. When we address a vertex, as $v_0 = v_0(i)$, for example, we indicate a number stored at the corresponding array index by using v_0 as a subscript. Here is the pseudo-code for calculating the set of coefficients $\{a_i^{l-1}\}$ from the set $\{a_i^l\}$ for the formula $a_i^{l-1} = (b^{l+1} * a^l)(m_3(i))$:

- **While** level $= l > 2$
- (First loop: accumulates at the vertices v_0 and v_1, the a_i^l values located at the points $m_1(i)$)
  - **For** $i = 0, i, <$ max$, i++, t_{v_0} \longleftarrow t_{v_0} + (1/4)a_i^l$; $t_{v_1} \longleftarrow t_{v_1} + (1/4)a_i^l$ **endFor;**
- (Second loop: accumulates the triangle vertex values at the m_3 points to form box spline coefficients $a_{m_3(i)}^{l-1}$):
  - **For** $i = 0, i <$ max$, i++, a_{m_3(i)}^{l-1} \longleftarrow a_{m_3(i)}^{l-1} + (1/4)t_{v_0} + (1/4)t_{v_1}$ **endFor;**
- Clear the set V_l vertex locations v_0 and v_1.
- (Third loop: subdivides the box spline coefficients $a_{m_3(i)}^{l-1}$ to the vertices v_0 and v_1):
  - **For** $i = 0, i <$ max$, i++ \widehat{t}_{v_0} \longleftarrow \widehat{t}_{v_0} + (1/4)a_{m_3(i)}^{l-1}$; $\widehat{t}_{v_1} \longleftarrow \widehat{t}_{v_1} + (1/4)a_{m_3(i)}^{l-1}$ **endFor;**
- (Fourth loop: subdivides the v_0 and v_1 values to form s_i^l and to store the difference d_i^l):
  - **For** $i = 0, i <$ max$, i++ s_i^l \longleftarrow (1/4)\widehat{t}_{v_0} + (1/4)\widehat{t}_{v_1}$; $d_i^l = a_i^l - s_i^l$; Store d_i^l at the point $m_1(i)$;
  - **endFor**

  (At this point we have stored the values d_i^l in the set C_l and the values a_j^{l-1} in the set C_{l-1}).
- Clear the set V_l for the next level: $l \leftarrow l - 1$;

endWhile

The result of this algorithm is that the coefficients d_i^l are stored in the sets C_l for $2 < l \leq 2m$. The flow of data is shown in Figure 2.

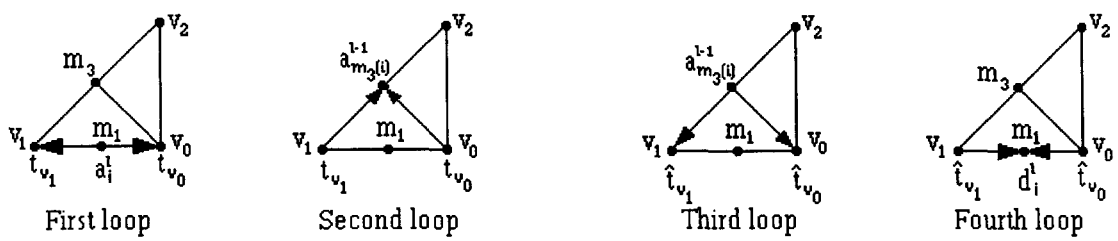

Figure 2.

The reconstruction of an image from the detail coefficients d_i^l is equally simple to implement. Note that during reconstruction, we may render a low resolution image at level l as a smooth or blurred image with no blocking effects by calculating the coefficients as $a_i^{l-1} = s_i^{l-1}$ (this is equivalent to setting $d_i^{l-1} = 0$). This is a kind of subdivision algorithm for rendering a box spline interpolation of the low resolution data points which are located at the level l lattice origin points. We may also note that the methods of [14,19] may be applied to the coefficients s_i^l to produce

edge detecting wavelet masks which are identical to the refinement masks of wavelets obtained by Ron and Shen [6].

3. CONCLUSION

We have looked at a one-dimensional pyramid algorithm for box spline multiresolution calculations via Sierpinski scanning. Many other similar algorithms are possible, but the one chosen here is particularly simple.

There is one obvious objection which the reader may have noticed. These are not the usual pyramid algorithms for computing a wavelet transform in which the wavelet coefficients are the coordinates of the image in a new orthogonal wavelet basis of the finite-dimensional space of images. Indeed, we are possibly storing twice as many coefficients as usual, namely, $2 \times 2^m \times 2^m$. This is to be expected since, in any four-directional box spline wavelet representation, the wavelets form not a basis but a redundant frame for $L_2(\mathbf{R}^2)$. If our goal is image compression, we may not be hampered in most cases since there will be many zero (or small) differences. We may also regain the bulk of the storage by using some other averaging such as the Haar wavelet transform to capture details at the the first few levels of fine resolution. The goal is to store as much energy as possible in the low resolution coefficients with a low level of computational complexity. The quadratic box splines offer this possibility.

For applications of the methods of this article to image processing, see [21].

REFERENCES

1. C.K. Chui, *An Introduction to Wavelets*, Academic Press, (1992).
2. E.P. Simoncelli and E.H. Adelson, Non-separable extensions of quadrature mirror filters to multiple dimensions, *Proc. IEEE* **78**, 652–663, (1990).
3. G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, (1996).
4. HyungJun Kim, Unified image compression using reversible and fast biorthogonal wavelets, Ph.D. Thesis, University of Pittsburgh, (1996).
5. C.K. Chui, K. Jetter and J. Stöckler, Wavelets and frames on the four-directional mesh, In *Wavelets, Theory, Algorithms, and Applications*, (Edited by C.K. Chui, L. Montejusco and L. Puccio), pp. 213–230, (1994).
6. A. Ron and Z. Shen, Compactly supported tight affine frames in $L_2(R^d)$, *Math. Comp.* (to appear).
7. A. Ron and Z. Shen, Affine systems in $L_2(R^d)$: Analysis of the analysis operator, *J. Funct. Anal.* (to appear).
8. P. Zwart, Multivariate splines with nondegenerate partitions, *SIAM J. Numer. Anal.* **10**, 665–673, (1973).
9. M.J.D. Powell and M.A. Sabin, Piecewise quadratic approximations on triangle, *SIAM J. Number. Anal.* **10**, 665–673, (1973).
10. H. Diamond, L.A. Raphael and D.A. Williams, Quasi-interpolant box-spline formulation for image compression and reconstruction, In *Wavelet, Images and Surface Fitting*, (Edited by P.J. Laurent, A. LeMehaute and S.L. Schumaker), pp. 221–228, A. K. Peters, Wellesley, MA, (1994).
11. I. Daubechies, Ten lectures on wavelets, In *CBMS-NSF Conference Series in Applied Mathematics*, Volume 61, SIAM, Philadelphia, PA, (1992).
12. A. Cohen and I. Daubechies, Non-separable bidimensional wavelet bases, *Rivista Mathematica* **9**, 51–137, (1993).
13. H. Sagan, *Space Filling Curves*, Springer-Verlag, (1994).
14. D.J. Hebert, Cyclic interlaced quadtree algorithms for quincunx multiresolution, *Journal of Algorithms* (to appear).
15. D.J. Hebert, Symbolic local refinement of tetrahedral grids, *J. Symbolic Computations* **17**, 457–472, (1994).
16. J.M. Maubach, Local bisection refinement for n-simplicial grids generated by reflections, *SIAM J. Sci. Comput.* **16**, 210–227, (1995).
17. H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*, Addison-Wesley, (1989).
18. D.J. Hebert and HyungJun Kim, Image encoding with triangulation wavelets, In *Proc. SPIE, Wavelet Applications in Signal and Image Processing*, Volume 2569, pp. 381–392, (1995).
19. D.J. Hebert and HyungJun Kim, A fast wavelet compass edge detector, In *Proc. SPIE, Wavelet Applications in Signal and Image Processing*, Volume 2825, pp. 432–442, (1996).
20. C. de Boor, K. Hollig and S. Riemenschneider, *Box Splines*, Springer-Verlag, New York, (1993).
21. D.J. Hebert and HyungJun Kim, Applications of binary triangulations and interlaced quadtrees in image processing (to appear).